Revista Ciencia & Desarrollo 2015; 19: 61-67 / ISSN 2304-8891

CLASIFICACIÓN DE DÍGITOS MANUSCRITOS DE IMÁGENES DIGITALES

THE SPECIES OF THE EUPHORBIACEAE FAMILY IN TACNA PROVINCE: BIOSYSTEMATIC STUDY

[1]**Carlos Alberto Silva Delgado,** [2]**Euler Tito Chura**

RESUMEN

En Ciencias de la Computación el reconocimiento de dígitos escritos a mano en imágenes digitales es de suma importancia, ya que a partir de esto, se pueden hacer distintas tareas, entre las que destacan el reconocimiento y reconstrucción de caracteres. Los algoritmos de clasificación tienen la capacidad de recuperar en su totalidad patrones aprendidos a partir de patrones de entrada, en este caso se utiliza como patrones de entrada los dígitos manuscritos de la Base de Datos del MNIST. En el presente trabajo se presenta la aplicación (Base de Datos del MNIST) de los algoritmos de clasificación de dígitos como parte aplicativa en el campo del Reconocimiento de Patrones, se hace una comparación entre la performance de los algoritmos de clasificación: SVM, Distancia Euclidiana, Vecino más cercano, J48. Los resultados demostraron que el clasificador SVM es el más eficiente para clasificar dígitos manuscritos con respecto a los otros clasificadores, por obtener un error del 1,04 %.

Palabras Clave: aprendizaje supervisado, clasificación, reconocimiento de patrones, reconocimiento de dígitos manuscritos.

ABSTRACT

In Computer Science the recognition of handwritten digits on digital images is paramount, from this they can do many tasks, among which are the recognition and reconstruction of characters. Classification algorithms have the ability to recover fully learned patterns from input patterns; in this case it is used as input patterns the digits manuscripts from MNIST Database. In this work (MNIST Database) application of the classification algorithms digits is presented as an applicative part in the field of pattern recognition, which presents a comparison between the performance of sorting algorithms: SVM, Euclidean distance, Nearest neighbor, J48; where the results demonstrated that the SVM classifier is the most efficient to classify handwritten digits compared to other classifiers, to obtain an error of 1,04 %.

Keywords: supervised learning, classification, pattern recognition, recognition of handwritten digits.

INTRODUCCIÓN

Aprender de los datos es una de las formas básicas que los seres humanos percibimos del mundo y, por lo tanto, adquirimos los conocimientos. Hoy en día, hay cantidad de datos disponibles y sorprendentemente van en aumento en el mundo de Internet y en aplicaciones industriales. Hay tareas de clasificación con un gran número de clases como el reconocimiento de caracteres, manuscritos chinos con más de 6000 clases. En un problema de categorización de documentos en la Web son procesados gigabytes de datos con alta dimensión (Dong *et al.*, 2003).

Un sistema de reconocimiento de patrones completo consiste en (Seijas, 2008):

- Un *sensor* que recoge las observaciones a clasificar.
- Un *sistema de extracción de características* que transforma la información observada en valores numéricos o simbólicos.
- Un *sistema de clasificación* o descripción que, basado en las características extraídas, clasifica la medición.

El punto esencial del reconocimiento de patrones es la clasificación. Por ejemplo, se puede clasificar imágenes digitales de letras en las clases «A» a «Z» dependiente de sus píxeles o se pueden clasificar huellas dactilares (Rodrigues *et al.*, 2007).

Clasificación

El problema de la clasificación es el más frecuente en la práctica, de esta forma, construiremos modelos que permita predecir la categoría de las instancias en función de una serie de atributos de entrada. La clase se convertirá en variable objetivo a predecir. La figura 1 muestra el proceso de preparación de datos.

Dominios de la aplicación.

En un artículo publicado (Palacios y Gupta, 2003) se indica que el volumen de cheques que deben procesar los bancos, conllevan unos costes que crecen cada año, pues alre-

[1] Ingeniero en Informática y Sistemas. Docente de la Carrera Profesional de Ingeniería de Sistemas de la Universidad Nacional de Moquegua. Moquegua – Perú.

[2] Maestro en Ciencias Computación e Informática, Ingeniero Estadístico. Docente de la Carrera Profesional de Ingeniería de Sistemas de la Universidad Nacional de Moquegua. Moquegua – Perú.

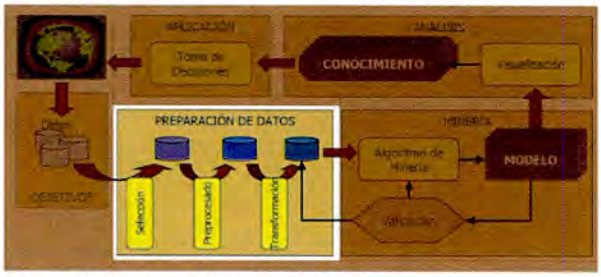

Figura 1. Preparación de los datos.

dedor de 50 millones de tales documentos se procesan anualmente en los Estados Unidos (Dong *et al.*, 2003), constituyendo en más del 60 % de los pagos que no se realizan en metálico, según un informe del Federal Reserve Bank .

Actualmente se están empleando en el Reconocimiento de placas en vehículos automotores, donde el objetivo principal es realizar un control vehicular mediante el Reconocimiento Óptico de Caracteres (OCR) de la Placa de un vehículo, utilizando una cámara USB y posteriormente procesarla (Legarda y Chacon, 2010; Andrade y Lopez, 2009).

Trabajos relacionados

En el trabajo de Ceballos (2007) se aplicó un método de clasificación de números manuscritos basado en prototipos creados usando la distancia euclidiana o error de datos utilizada. Consta de 500 imágenes de números (de 20 x 20 pixeles) para entrenamiento y 200 para validación con sus respectivas salidas etiquetadas entre 0 y 9, obtenidas de la base de datos MNIST. También, este método fue comparado con un algoritmo de redes neuronales perceptrón multicapa (MLP), con entrenamiento supervisado backpropagation y funciones de activación sigmoidal, tipo "logsig". La mejor clasificación lograda con esta base de datos fue de 80 %. Este porcentaje de clasificación correcta fue bastante similar al obtenido con la red neuronal (82 % aproximadamente en el mejor caso).

Vicente *et al.* (2004) proponen un método de clasificación de números manuscritos basado en distancia Euclidiana a prototipos. Los resultados con la base de prueba, con clasificación mediante el prototipo más cercano, alcanzaron un 93,5 % de clasificación correcta, utilizando un umbral de creación u=7,7. Cabe destacar que, utilizando el mismo umbral de creación (u=7,7) el método con desplazamiento mejora el desempeño del sistema, ya que mejora el porcentaje de clasificación correcta (90,1 % a 93,5 %) y disminuye al mismo tiempo el número de prototipos (250 a 184). Si el entrenamiento se hace con un umbral de creación u más bajo (u=4,5) y con desplazamiento, aumenta la cantidad de prototipos a 267. Clasificando mediante una función de votación lineal y distancia de aceptación d_a=0,6, se logra un reconocimiento correcto de 94,8 %. Estos resultados fueron comparados, utilizando los mismos patrones de entrenamiento y prueba, con una red perceptrón multicapa entrenada con patrones desplazados hasta el borde, logrando un desempeño de 91,8 %, y con un algoritmo SOM+LVQ1, también entrenado con desplazamiento hasta el borde, logrando un 91,5 %.

Dong *et al.* (2003) emplearon el algoritmo del Support Vector Machine (SVM), donde integra el kernel caching, digest y shrinking y obtienen un error del 0,6 %, trabajando con la base de datos del MNIST con un preprocesamiento de las 28 x 28 (784), características iniciales efectuadas por el grupo de investigación encabezado por LeCun, quien mediante una transformación lineal, tuvo una imagen resultante de gray-level e hizo también la reducción de la dimensión a 576, por generar un vector, cuya característica es de 6 horizontal, 6 vertical y 16 "*directional resolutions*".

En Martínez (2006) la clasificación de imágenes de caracteres se realizó mediante el algoritmo de clasificación J48, obteniendo aceptables resultados.

Misukami y Tadamuera (2002) utilizan el algoritmo del vecino más cercano, obteniendo un error del 0,57 % con la base de datos MNIST de los dígitos manuscritos. El tamaño de la imagen originalmente fue de 28 × 28 pixeles, pero se extendió a 32 × 32 pixeles por relleno de una región de cuatro pixeles. El número de la etapa de la estrategia de gruesa a fina se establece en 3, y los tamaños de imágenes en cada etapa fueron de 8 × 8, 16 × 16, y 32 × 32 pixeles, respectivamente.

Estado del arte

El reconocimiento óptico de caracteres (OCR) es un área de investigación importante en reconocimiento de patrones. El objetivo de un sistema OCR es reconocer letras del alfabeto, otros caracteres y, en nuestro caso, dígitos que, capturados en forma de imágenes digitales, sean reconocidos sin ninguna intervención humana.

La escritura de cada persona es diferente; por lo que es necesario utilizar una base de datos que contenga un gran número de dígitos escritos a mano. En este trabajo se clasifican imágenes dependiendo de sus características de los manuscritos, usando el conjunto de datos MNIST (http://yann.lecun.com/exdb/mnist/). El MNIST dataset consiste de 60000 imágenes de los diez dígitos, el mismo que se usará como Training. El dataset del Testing tiene un conjunto independiente de 10000 dígitos que obviamente es para evaluar la performance de los clasificadores. Alguna muestra de los dígitos del MNIST se puede apreciar en la figura 2.

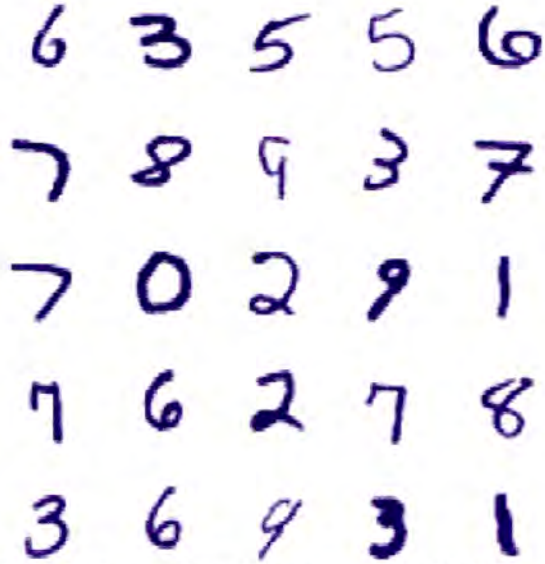

Figura 2. Muestra de los dígitos manuscritos MNIST.

MATERIALES Y MÉTODOS

Método de extracción de características

Nuestro objetivo es el reconocimiento de imágenes de dígitos manuscritos basado en métodos de clasificación de una dataset multivariable, donde las *variables* son los *valores de los pixeles* y las *clases* son *cada tipo de dígito.*

El dataset consta de 28x28 = 784 pixeles de imágenes en *grey scale* de dígitos en el rango de 0-9, como se puede ver en la figura 3.

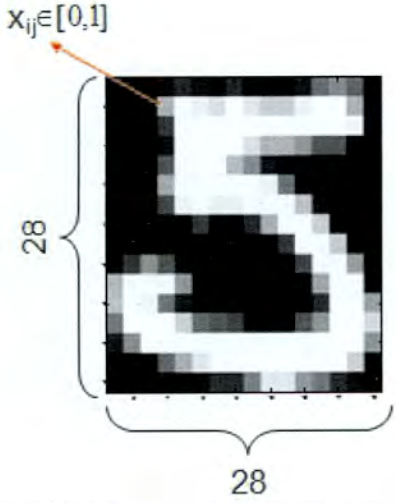

Figura 3. Dígito representado en 28x28 pixeles.

Además:

$$X_i=[x_{i1}, x_{i2}, \ldots\ldots x_{i784}]$$
$$y_i \in \{0,1,2,3,4,5,6,7,8,9\}$$

Para la extracción de características, se ha elegido cómo método el *Análisis de Componentes Principales* (PCA). El objetivo del análisis es representar las medidas numéricas de un conjunto de variables en un espacio de dimensión mucho menor (Vicente *et al.*, 2004). Esta representación debe ser tal, que al eliminar dimensiones superiores, la pérdida de información sea mínima. Como vimos en nuestro caso son 28 x 28 pixeles por cada imagen, entonces, harían una dimensión de 784. Realizar una operación con gran cantidad de vectores de esta magnitud sería altamente costoso computacionalmente, además de la existencia de variables (características) correlacionadas. Este método de acuerdo a lo revisado en la bibliografía sobre la reducción de dimensión y extracción de características es lo que se llama también Eigenvalues o Eigenvectors, nuestros dígitos que será nuestro dataset de entrada, lo analizaremos con el PCA.

Implementación del PCA

Para el proceso de entrenamiento hemos recurrido a la herramienta de software Matlab. Se ha tenido que realizar los siguientes pasos:

a) Llevar a forma de vector cada imagen (dígito) de entrenamiento; para ello se ha tenido que extraer los datos de la base de datos (que están en binario) del MNIST, con el siguiente script en Matlab.

```
function [I,labels,I_test,labels_test] =
readMNIST(num)
path = 'train-images.idx3-ubyte';
fid = fopen(path,'r','b');
magicNum = fread(fid,1,'int32');
if(magicNum~=2051)
    display('Error: cant find magic number');
    return;
end
imgNum = fread(fid,1,'int32');
rowSz = fread(fid,1,'int32');
colSz = fread(fid,1,'int32');
if(num<imgNum)
    imgNum=num;
end
for k=1:imgNum
        I{k} = uint8(fread(fid,[rowSz
colSz],'uchar'));
end
fclose(fid);
path = '.\MNIST\train-labels.idx1-ubyte';
fid = fopen(path,'r','b');
magicNum = fread(fid,1,'int32');
if(magicNum~=2049)
    display('Error: cant find magic number');
    return;
end
itmNum = fread(fid,1,'int32');
if(num<itmNum)
    itmNum=num;
end
labels = uint8(fread(fid,itmNum,'uint8'));
fclose(fid);
path = 't10k-images.idx3-ubyte';
fid = fopen(path,'r','b');
magicNum = fread(fid,1,'int32');
if(magicNum~=2051)
    display('Error: cant find magic number');
    return;
end
imgNum = fread(fid,1,'int32');
rowSz = fread(fid,1,'int32');
colSz = fread(fid,1,'int32');
if(num<imgNum)
    imgNum=num;
end
 for k=1:imgNum
        I_test{k} = uint8(fread(fid,[rowSz
colSz],'uchar'));
end
fclose(fid);
 path = 't10k-labels.idx1-ubyte';
fid = fopen(path,'r','b');
magicNum = fread(fid,1,'int32');
if(magicNum~=2049)
    display('Error: cant find magic number');
    return;
end
itmNum = fread(fid,1,'int32');
if(num<itmNum)
    itmNum=num;
end
labels_test = uint8(fread(fid,itmNum,'uint8'));
fclose(fid);
```

Script 1: Extracción del dataset MNIST (dígitos manuscritos) en Matlab

Luego de ejecutar el script de MatLab nos devuelve un vector de 60000 matrices (training), cada una de las matrices tiene una dimensión de 28 x 28 por cada dígito, y en el caso del testing, de la misma forma.

Posterior a este proceso se tuvo que separar cada una de las imágenes en una determinada matriz de dimensión 60000x784 (training), es decir, se ha linealizado la matriz que contiene cada una de las imágenes de los números. Ahora ya tenemos cada dígito (vector) con 784 características y lo mismo con el dataset del testing de dimensión 10000x784; obteniendo el siguiente reporte. (Tabla 1).

Tabla 1. BD MNIST de los dígitos manuscritos.

Dígito	Training	Testing
0	5923	980
1	6742	1135
2	5958	1032
3	6131	1010
4	5842	982
5	5421	892
6	5918	958
7	6265	1028
8	5851	974
9	5949	1009
TOTAL	60000	10000

b) Reducción de la Dimensionalidad en la BD original.

Inicialmente en éste trabajo como método de simplificación y reducción de la dimensionalidad (Vicente *et al.*, 2004) se ha utilizado el Análisis de Componentes Principales (PCA), puesto que 784 (28x28) características contiene mucho ruido, entonces, se empleó PCA en el training y testing dataset porque:

- Ambos datasets (training y testing) deben estar en el mismo dominio (PCA).
- Ambos datasets deben tener la misma dimensionalidad. Hay casos muy particulares donde esto no se aplica, pero este no es el caso.

Las 28 x 28 características de los dataset están representadas por: C1, C2, ..., C784. Inicialmente presentamos el Gráfico de Sedimentación (figura 4) de los dataset para observar el comportamiento de los componentes y visualizar a priori el número de componentes a tomar en cuenta como dataset en los clasificadores.

El código en Matlab que nos permitió procesar el PCA fue el siguiente:

```
X=X';    %transpuesta de la entrada de la matriz de
X, ahora cada fila es un elemento
[l,N]=size(X);
% Sustrayendo la media
mean_vec=mean(X')';
X_zero=X-mean_vec*ones(1,N);
% Calculando la covarianza y sus %eigenva-
lues/eigenvectors
R=cov(X_zero');
[V,D]=eig(R);

eigenval=diag(D);
[eigenval,ind]=sort(eigenval,1,'descend');
eigenvec=V(:,ind);

explain=eigenval/sum(eigenval);
%Guardando los primeros m %eigenvaules / eigen-
vectors
eigenval=eigenval(1:m);
eigenvec=eigenvec(:,1:m);
% Calculando la transformación de la matriz
A=eigenvec(:,1:m)';
% Calculando la transformación del dataset
Y=A*X;
Y=Y';  %Se transpone para que cada fila sea un
elemento vector
```

Script 2: Implementación del Algoritmo del PCA en Matlab

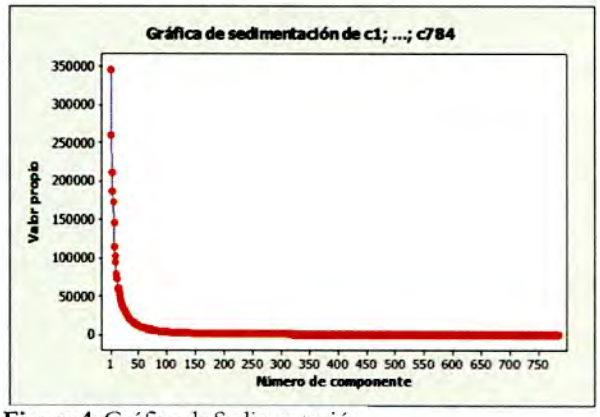

Figura 4. Gráfico de Sedimentación.

Tabla 2. Resultados del PCA de 28x28 pixeles

Componente	Eigenvalue	% total de Varianza	% Acumulado
1	345318	10%	10%
2	259289	7,5%	17,5%
3	211037	6,1%	23,7%
...	...	...	...
...	...	...	...
623	2	0%	100%
624	2	0%	100%
625	1	0%	100%
626	1	0%	100%

El algoritmo del PCA implementado en Matlab nos permitió aplicar la transformación lineal, obteniéndo los resultados de autovalores (eigenvalues) y autovectores (eigenvectores) y la transformación del dataset, información que presentamos necesaria en la tabla 2, a fin de tomar la decisión del número de componentes a considerar.

La interpretación de la tabla 2 es la siguiente:

- El primer componente principal explica el 10 % de la variación total en los datos.
- El segundo componente explica el 7,5 % de la variación total de los datos.
- Los tres primeros componentes principales con varianzas iguales a los eigenvalues representan el 23,7 % de la variabilidad total.

Según lo demostrado formalmente por Misukami y Tadamuera (2002), debemos conservar los componentes principales con valores propios (eigenvalues) mayores que 1, es decir, hemos tomado los primeros *624 componentes principales.*

Reporte de los clasificadores de la BD original

En primera instancia ejecutamos los scripts en Matlab de los Clasificadores, como se aprecia en la tabla 3, los resultados no son muy alentadores con el uso de los datos originales del MNIST (28x28), lo cual demuestra que aún aplicando el PCA a ese dataset tenemos mucho ruido, por lo que nos vemos obligados a recurrir a otra estrategia.

Tabla 3. Resultados de Clasificadores con PCA en 28x28 pixeles.

Clasificador	Error
SVM (Kernel Gaussiano)	54,45%
J48	57,25%
Distancia Euclidiana	58,00%
Vecino más cercano k=1	65,17%

Como *las tasas de errores son muy altos*, y siendo nuestro objetivo minimizar la tasa de error de los Algoritmos de Clasificación, usaremos los dataset obtenidos por el grupo de investigación encabezado por LeCun (Dong *et al.*, 2003) mediante una transformación lineal. Producto de ello se obtuvo una imagen resultante de gray-level y se hizo también la reducción de la dimensión a 576 (de 784) por generar un vector características de 6 horizontal, 6 vertical y 16 "*directional resolutions*". Por lo tanto experimentaremos la nueva BD con 576 características de los dígitos manuscritos.

Aplicando PCA a las nuevas 576 características de la BD de dígitos manuscritos MNIST, con ayuda del Script 02 (Matlab), se reduce a 92 componentes principales, cuyo reporte se muestra a continuación. (Tabla 4).

Tabla 4. Resultados del PCA de 24x24 pixeles.

Componente	Eigenvalue	% total de Varianza	% Acumulado
1	0,28854	19%	19%
2	0,17524	11,6%	30,6%
3	0,14666	9,7%	40,3%
...	...	...	...
...	...	...	...
...	...	...	..
91	0,00082	0,1%	96,5%
92	0,00081	0,1%	96,6%
93	0,00080	0,1%	96,7%
94	0,00079	0,1%	96,7%

La interpretación de la tabla 4 *(ordenados según el Eigenvalue, del más importante al menos importante)* es la siguiente:

- El primer componente principal explica el 19 % de la variación total en los datos.
- El segundo componente explica el 11,6 % de la variación total de los datos.
- Los tres primeros componentes principales con varianzas iguales a los eigenvalues representan el 40,3 % de la variabilidad total.

Vemos que la varianza capturada es el 96,5 %, entonces, el nuevo dataset de componentes principales fueron procesados con los algoritmos implementados en Matlab.

RESULTADOS

Evaluación de la performance

El primer programa implementado en Matlab fue el *Algoritmo J48* (Misukami y Tadamuera, 2002), que tuvo como entrada el dataset de 92 componentes principales y obtuvo una tasa de error del 5,12 %.

Seguidamente se ha implementado también en Matlab el *Algoritmo de clasificación de la distancia euclidiana*, logrando reducir la tasa de error a 1,15 %.

Aplicando la misma data al *Algoritmo del vecino más cercano*, se obtuvo un error del 1,15 %.

Finalmente, se ha procesado la data con el algoritmo del *Support Vector Machine*, también en Matlab logrando mejorar el error a 1,04 %.

En las tablas del 5 al 8 se muestra un resumen de los algoritmos implementados, la misma que podemos visualizar en la tabla 9, ordenados de acuerdo a la tasa de error.

En la tabla 9 se observa que se reduce drásticamente las tasas de error de los mismos clasificadores empleados con la dataset original (28x28 pixels), obteniendo la menor tasa de error de 1,04 % con el algoritmo SVM.

Cabe indicar que las características de la computadora donde se ejecutó los algoritmos implementados fueron las siguientes:

- 4 GB de memoria RAM,
- Procesador Intel CORE 2
- MatLab R2010a
- Sistema operativo Windows 7

Tabla 5. Matriz de Confusión del Algo*ritmo J48*.

	0	1	2	3	4	5	6	7	8	9
0	957	0	4	2	3	2	5	0	6	1
1	0	1118	5	0	2	0	3	1	6	0
2	8	6	969	18	3	3	3	8	10	4
3	1	1	14	961	1	14	2	5	4	7
4	4	5	9	0	927	3	6	5	7	16
5	5	3	6	15	0	835	6	2	18	2
6	12	2	4	2	4	8	915	0	11	0
7	1	5	11	7	5	0	0	979	5	15
8	8	3	8	6	14	12	15	0	889	14
9	2	4	5	7	18	6	0	0	18	938

Aciertos: 94,88 %

Desaciertos 5,12%

Tabla 6. Matriz de Confusión del Algoritmo de la Distancia Euclidiana.

	0	1	2	3	4	5	6	7	8	9
0	978	0	0	0	0	0	2	0	0	0
1	0	1131	1	0	0	0	2	1	0	0
2	1	1	1023	0	0	0	0	5	2	0
3	0	0	3	999	1	2	0	4	1	0
4	0	0	1	0	969	0	3	0	1	8
5	1	0	1	11	1	875	2	0	1	0
6	2	1	0	0	0	0	953	0	2	0
7	0	3	4	1	2	0	0	1012	0	6
8	2	0	1	1	4	2	1	1	956	6
9	0	3	0	0	7	2	0	4	3	989

Aciertos: 98,85%

Desaciertos 1,15%

Tabla 7. Matriz de Confusión del Algoritmo Vecino más cercano.

	0	1	2	3	4	5	6	7	8	9
0	977	0	0	0	0	0	2	0	0	0
1	0	1132	1	0	0	0	2	1	0	0
2	1	1	1022	0	0	0	0	5	2	0
3	0	0	3	1000	1	2	0	4	1	0
4	0	0	1	0	969	0	3	0	1	8
5	1	0	1	11	1	875	2	0	1	0
6	2	1	0	0	0	0	954	0	2	0
7	0	3	4	1	2	0	0	1011	0	6
8	2	0	1	1	4	2	1	1	957	6
9	0	3	0	0	7	2	0	4	3	988

Aciertos: 98,85 %

Desaciertos 1,15%

Tabla 8. Matriz de Confusión del Algoritmo SVM.

	0	1	2	3	4	5	6	7	8	9
0	977	0	0	0	0	0	2	1	0	0
1	0	1132	2	0	0	0	1	0	0	0
2	1	1	1022	0	2	0	0	5	1	0
3	0	0	2	1004	0	1	0	2	1	0
4	1	1	3	0	966	0	2	1	3	5
5	1	0	1	6	0	879	1	1	1	2
6	5	1	1	0	2	5	943	0	1	0
7	0	2	4	1	2	0	0	1019	0	0
8	1	0	1	0	2	2	1	1	963	3
9	0	2	0	2	7	0	0	1	6	991

Aciertos: 98,96%

Desaciertos 1,04%

Tabla 9. Resultados de Clasificadores con PCA en 24x24 pixeles.

Clasificador	Error	Tiempo de Ejecución
SVM (Kernel Gaussiano)	1,04%	3,17 minutos
Vecino más cercano k=1	1,15%	0,71 minutos
Distancia Euclidiana	1,15%	0,76 minutos
J48	5,12%	13,52 minutos

DISCUSIÓN

Recuperando la Base de Datos MNIST en formato de imágenes con la ayuda de Matlab, vemos que incluso los seres humanos NO podríamos clasificar dichos dígitos, listamos algunos ejemplos que fueron clasificados con el algoritmo SVM y el Vecino más cercano (K-nn).

¿Cómo podría ser mejorado el sistema?

- Mejorar el Sistema de Clasificación significa minimizar la tasa de error, por lo tanto se tendría que probar resultados con otros algoritmos tales como: Redes Neuronales, los

Tabla 10. Ejemplos donde el reconocimiento no funciona correctamente.

Dígito (testing)	Clase del MNIST	Clasificador	
		SVM	K-nn
	2	1	7
	1	7	1
	8	9	8
	9	5	9
	9	7	9
	5	3	5
	8	2	3
	9	4	9
	5	3	5
	9	4	9
	4	9	4
	2	0	2
	5	3	5
	6	1	6
	4	9	4
	0	6	0
	6	0	6
	6	8	6

Dígito (testing)	Clase del MNIST	Clasificador	
		SVM	K-nn
	1	7	9
	1	7	1
	9	4	9
	0	5	0
	5	3	5
	7	2	7
	2	7	8
	5	6	5
	0	6	0

diversos variantes de algoritmos Bayesianos, que también son utilizados para clasificación.

- Una etapa importante también sería aumentando el dataset de dígitos manuscritos con más imágenes

CONCLUSIONES

El problema de clasificación de imágenes, en específico de dígitos manuscritos ha sido abordado desde multitud de enfoques por la comunidad científica, por lo que en el presente trabajo se ha querido demostrar y corroborar la eficiencia de los algoritmos clasificadores utilizados en los artículos citados debido a su performance con los dígitos manuscritos y que han sido mejorados por investigadores en el área, con el objeto de minimizar el porcentaje de error, no obstante, sigue siendo un desafío en la ciencia de la computación, donde es muy importante tener la destreza técnica de las herramientas software en la implementación de los algoritmos, cuando se procesan grandes bases de datos y evitar el desbordamiento de memoria.

En *primera instancia* procesando el dataset con las 28x28 características se han obtenido tasas de errores muy altos, pese a la aplicación del PCA (tabla 3).

En *segunda instancia* se ha procesado el dataset de 24x24 características con previa reducción de la dimensión con el método PCA, mejorando notablemente las tasas de error (tabla 9).

Se ha demostrado (tabla 9) la eficiencia del clasificador Support Vector Machine (SVM) con Kernel Gaussiano, con respecto a otros clasificadores, porque se ha obtenido un error del 1,04 % vs los algoritmos del Vecino más cercano y la Distancia Euclidiana, ambos con 1,15 % de error.

REFERENCIAS BIBLIOGRÁFICAS

Andrade, G., Lopez, J. (2009). Sistema de Control Vehicular utilizando OCR. *Publicaciones de Escuela Superior Politécnica de Litoral. Ecuador*.

Ceballos, G. (2007). Clasificación de Números Manuscritos Basados en Prototipos Creados usando la Distancia Eucliciana como Umbral. *Publicaciones Dpto de Ing. Eléctrica. Universidad de Chile*, 5-6.

Dong, J. X., Krzyzak, A., Suen, C. Y. (2003). A fast SVM training algorithm. *International Journal of Pattern Recognition and Artificial Intelligence, 17*(3), 367-384.

Legarda, A., Chacon, M. (2010). Localización de placas de Vehículos Automotores. *Publicaciones Instituto Tecnológico de Chihuahua. México*, 7-9.

Mayta, W. (2008). Reconocimiento Automático de dígitos manuscritos en base a prototipos multivalidados. *Publicaciones Universidad Mayor de San Andrés*, 2-3.

Misukami, Y., Tadamuera, K. (2002). CUDA implementations of deformable Pattern Recognition and its application to MNIST hardwritten digit database. 15-17.

Martínez, G. (2006). Clasificación mediante conjuntos. *Publicaciones Universidad Autónoma de Madrid*, 6-8.

Palacios, R., Gupta, A. (2003). Sistemas de Reconocimiento de Caracteres para la lectura automática de Cheques. *Publicaciones Massachussets Institute of Tecnology*, 18-20.

Rodrigues, R., Viana, R., Pasquali, A. Pistori, H. (2007). Máquinas de Vetores de Suporte Aplicadas à Classificação de Defeitos em Couro Bovino. *Publicaciones Universidade Católica Dom Bosco*, 2.

Seijas. L. M. (2008). Reconocimiento de dígitos manuscritos mediante Redes Neuronales: Una Técnicas Híbrida. *Publicación Semestral Dpto de Computación*, Universidad de Buenos Aires, 3-4.

Vicente, M.A., Fernandez, C. Puerto, R., Gil, A. (2004). Extracción de características para reconocimiento visual. *Actas de las XXV Jornadas de Automática*. ISBN: 84-688-7460-4.

Correspondencia:

Carlos Alberto Silva Delgado: *calbertosilvad@hotmail.com*

Hugo Euler Tito Chura: *eulertito@hotmail.com*

Fecha de Recepción: 24/03/2015

Fecha de Aceptación: 01/06/2015